Représentation optimiste de contenus dans les systèmes P2P

Anthony Ventresque[1,2], Philippe Lamarre[1,2], Sylvie Cazalens[1], and Patrick Valduriez[2,3]

[1]LINA, Université de Nantes
FirstName.LastName@univ-nantes.fr
[2]INRIA
FirstName.LastName@inria.fr
[3]LIRMM, Université de Montpellier

Résumé Nous nous intéressons aux systèmes distribués et hétérogènes disposant de nombreuses sources d'information autonomes, tels que les systèmes pair à pair non structurés. Dans de tels systèmes, le routage des requêtes, en particulier les requêtes top-k, est souvent limité à des algorithmes simples proches des parcours en largeur ou en profondeur, tels le flooding. Des améliorations utilisant des processus de gossiping ou des historiques des requêtes passées ont été proposées. Cependant, ils restent gourmands en envois de messages et ne garantissent pas d'obtenir les meilleurs résultats. En effet, ils ne disposent pas d'une représentation des sources leur permettant de décider a priori si l'envoi d'une requête est inutile. Dans ce but, nous définissons la propriété d'optimisme de l'évaluation d'une représentation d'une source par rapport à une mesure de pertinence : celle-ci, pour une requête donnée, évalue toujours la représentation d'une source avec une valeur supérieure ou égale à celle qu'aurait n'importe quelle information accessible via cette source, à une distance donnée. Dans un premier temps, nous motivons l'utilité d'une telle représentation sur des exemples. Nous proposons ensuite une analyse des propriétés nécessaires à la représentation d'un ensemble de documents pour l'usage visé. Puis,dans le cas de documents représentés par des vecteurs sémantiques, nous définissons (i) une agrégation incrémentale et composable et (ii) une représentation optimiste par rapport à une mesure de pertinence basée sur le cosinus.

1 Introduction

Dans les réseaux pairs-à-pairs (P2P) non structurés, le routage de requêtes est très souvent gourmand en ressources : puisqu'il n'est pas possible pour un pair de localiser précisément les sources d'information pertinentes, il lui faut interroger tout son voisinage. Concernant les requêtes k-meilleurs (top-k), où il est demandé de retourner les k résultats les plus pertinents, la technique par inondation du voisinage (*flooding*) est très utilisée, bien qu'elle nécessite beaucoup de ressources (messages, calculs), et connaît finalement peu d'améliorations. Certaines solutions proposent de n'envoyer qu'à certains voisins, en tenant compte de résultats antérieurs sur des requêtes similaires [10,18] ou d'estimations par gossiping sur les contenus des pairs [5,22]. Mais ces techniques, si elles limitent les envois de messages, n'assurent absolument pas d'avoir les meilleurs résultats. Il reste donc à définir des algorithmes qui fournissent les k-meilleures réponses à une distance déterminée de l'initiateur de la requête (par le TTL) tout en permettant de guider le routage et même de diminuer les envois.

Selon nous, le problème central est de faire en sorte que les pairs disposent d'une représentation satisfaisante de leurs voisins qui leur permette de juger l'intérêt d'un envoi. Pour cela, nous proposons de définir une représentation de pair *optimiste* : son évaluation de pertinence par rapport à la requête est toujours supérieure ou égale à celle de n'importe quelle information accessible via ce pair, ce pour une profondeur donnée. Ainsi, lors du routage, il serait possible de savoir si un voisin est susceptible de remonter des résultats plus intéressants que ceux qui sont déjà connus. Et donc s'il est nécessaire de lui envoyer la requête.

L'objectif de cet article est (i) de définir de manière générale des propriétés de la représentation d'un ensemble d'information utiles à une telle approche, (ii) proposer une représentation possédant ces propriétés dans le cas où les informations sont représentées par des vecteurs sémantiques.

Dans un cadre général, l'ensemble des informations accessibles via un pair à une profondeur donnée doit être représenté de manière synthétique. Ainsi, l'espace de représentation de l'ensemble n'est pas forcément le même que celui d'une information. Cependant, s'il semble naturel de distinguer plusieurs niveaux de représentation, au final, la pertinence doit être mesurée dans le même espace. Nous introduisons donc des fonctions d'aggrégation et de concrétisation de représentation qui nous permettent d'exprimer de manière précise la propriété d'optimisme. Par ailleurs, il faut prendre en compte l'évolution d'un ensemble d'information. En particulier, un ensemble ayant été agrégé, il est souhaitable que l'ajout d'une information ne nécessite pas le calcul complet de l'agregation du nouvel ensemble obtenu, mais profite plutôt de l'agrégation déjà calculée. Nous appelons cette propriété incrémentalité. De même la composabilité concerne deux agrégations que l'on veut fusionner.

Les espaces vectoriels sémantiques [21], sont un modèle de représentation utilisant les concepts d'une ontologie plutôt que des termes pour indexer chaque dimension. Chaque information est représentée par un *vecteur sémantique*. Ce type de représentation est très adapté à des documents et requêtes textuels par exemple. Chaque concept de l'ontologie est pondéré dans le vecteur sémantique

d'un document d_i (respectivement d'une requête q) suivant son importance dans d_i (resp. dans q).

Dans ce contexte, nous proposons de définir une fonction de représentation qui soit *optimiste* par rapport à la mesure de pertinence classique : le cosinus. Nous utilisons pour y arriver une fonction d'agrégation *incrémentale* et *composable*, ce qui permet de gérer la dynamique des agrégations déjà réalisées sans problème. Notre représentation d'un ensemble (exemple : un pair) permet d'assurer que pour une requête donnée, il n'est pas possible de trouver une partie (exemple : un document) ayant une valeur de pertinence plus forte que l'ensemble. Nous montrons que notre fonction est optimiste dans un cadre homogène. Nous explorons ensuite la possibilité d'une telle approche dans un cadre hétérogène.

Dans la section 2, nous motivons l'utilité de la propriété d'optimisme sur des exemples top-k. Nous définissons ensuite dans la section 3, les propriétés d'optimisme, d'incrémentalité et de composabilité dans un cadre général. Puis, dans le cas de vecteurs sémantiques, nous définissons les fonctions d'agregation et concrétisation et montrons les trois propriétés citées précédemment. La section 5 pose le problème du cadre hétérogène. Nous nous situons par rapport aux approches classiques dans la section 6, puis nous concluons (section 7).

2 Motivations

Nous considérons des systèmes P2P non-structurés qui permettent la plus grande autonomie des pairs et une plus grande tolérance aux fautes [3]. Nous nous intéressons au problème du *routage de requête* dans les systèmes P2P non-structurés, c'est-à-dire au problème de la diffusion de requêtes jusqu'aux pairs possédant des données pertinentes et à la remontée de leurs résultats [19]. Nous nous concentrons sur un seul type de requête, les requêtes top-k, dont l'utilité a déjà été montrée dans des domaines tels que la recherche d'information [12], les bases de données multimédia [8], la gestion de données en général lorsque le nombre de résultats est potentiellement très grand : le sous-ensemble des k meilleurs résultats est en général suffisant [2].

Par exemple, imaginons quatre pairs Alice, Bob, Charlie et Dave. Alice a deux voisins : Bob et Charlie; Charlie a un voisin, Dave (cf. figure 1). Le voisinage est dans notre modèle une relation non symétrique : Bob est voisin d'Alice qui n'est pas forcément sa voisine. Alice est initiatrice d'un envoi de requête dans le système. L'idée générale des algorithmes de routage dans les systèmes P2P est qu'Alice envoie sa requête à Bob et à Charlie (de manière séquentielle ou parallèle), puis que Charlie envoie la requête à Dave. Alice reçoit au plus k réponses des pairs qu'elle a contacté (moins de k s'ils ne disposent pas d'autant de réponses).

Par défaut, ces algorithmes n'utilisent aucune heuristique lors de l'envoi des requêtes, qui peut être séquentiel, parallèle ou hybride. Ainsi, ils ne savent pas s'ils peuvent obtenir des informations pertinentes via un pair donné. Le choix des pairs à qui envoyer lors d'un envoi séquentiel, parallèle ou hybride est donc

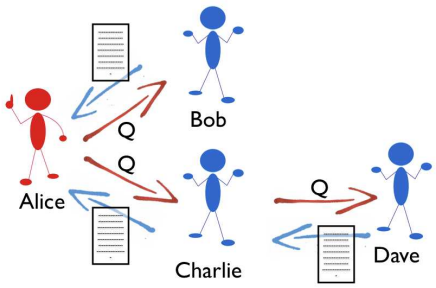

Fig. 1. Routage des requêtes et retour des k-meilleurs (ou moins de k) résultats classés par chacun des pairs.

aléatoire ; ce qui est a priori dommageable car une connaissance du voisinage permet de guider les envois de requêtes vers les pairs les plus pertinents. Certaines solutions utilisent donc des informations supplémentaires pour avoir une estimation de la pertinence d'un pair ; par exemple des historiques des requêtes antérieures, comme dans *intelligent BFS* [10] ou *adaptative probabilistic search* [18] ; mais aussi une indexation du contenu des voisins [5,22].

Prenons *intelligent BFS* [10]. Cette amélioration de Gnutella (algorithme d'inondation « en largeur d'abord ») utilise dans chaque pair une table de correspondances requête-pair pour tous ses voisins, indiquant les requêtes auxquelles les pairs ont répondu (c'est-à-die qu'ils avaient un résultat pertinent). Cet algorithme travaille sur des requêtes *exact match* où la fonction de pertinence est booléenne (vrai/faux). Dans ce cas, si un pair envoie une réponse à une requête donnée, c'est qu'il dispose d'une réponse pertinente à cette requête. Si une requête similaire doit à nouveau être routée par le pair en utilisant *intelligent BFS*, il interroge la table de correspondance afin de sélectionner un sous ensemble m de ses n voisins (paramétrable) à qui envoyer la requête. Cette solution diminue grandement le nombre de messages échangés, mais n'assure pas d'obtenir toutes les réponses. Selon nous, il manque à cette solution une représentation d'un pair qui permette de dire si effectivement il peut être intéressant de lui envoyer la requête. C'est ce que nous appelons une représentation *optimiste* : elle assure que son évaluation par rapport à la requête est forcément supérieure à celle de chacune des informations que l'on peut obtenir.

De même dans le cadre de requêtes k-meilleurs, il est aussi possible d'utiliser une représentation optimiste pour router. Puisque cette dernière nous donne la valeur maximale qu'une requête peut obtenir chez un pair ou dans un voisinage (un voisin et ses propres voisins jusqu'à un pas n), il est possible de réaliser des coupes dans les envois si un voisin ou un voisinage ne peut pas disposer de résultat intégrable dans la liste k-meilleurs actuelle.

Soit l'exemple de la figure 2 où Alice a trois voisins : Bob, Charlie et Eve. Nous supposons que ce sont des voisins terminaux, qui n'ont pas à transférer

la requête : soit le TTL vaut 1, soit ce sont des feuilles dans le réseau. Nous pouvons imaginer qu'Alice dispose d'une représentation du contenu de ses voisins lui permettant de guider son envoi. Ainsi elle enverrait à Bob avant Charlie et Eve, car celui-là a une valeur de pertinence plus élevée que ces deux-ci pour sa requête. Cette solution peut avoir au moins deux avantages. Le premier est que les meilleurs résultats pourraient être atteints les plus tôt. Envoyer à Bob, puis à Charlie et Eve, si celui-là est effectivement le plus intéressant, permet d'obtenir l'information la plus intéressante plus tôt.

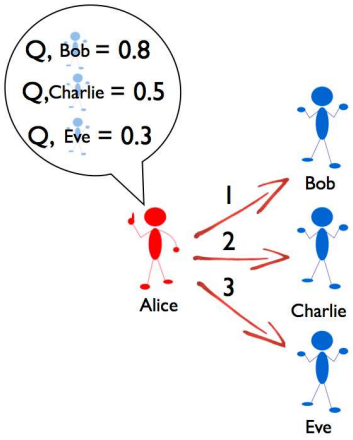

Fig. 2. La valeur de pertinence de Bob étant plus forte, Alice lui envoie sa requête en premier.

Imaginons de plus qu'il n'existe pas de donnée d'Eve qui ait une valeur de pertinence supérieure à celle de sa *représentation*. Cela nous donne une estimation haute du rang possible de ses résultats, i.e. de la valeur maximale dans le classement k-meilleurs qu'ils peuvent obtenir. Dans l'exemple, la valeur 0.3 qu'obtient la représentation d'Eve indique que cette dernière ne peut pas avoir de document ayant une meilleure valeur de pertinence. Supposons maintenant que la valeur la plus basse des résultats collectés par Alice dans son classement k-meilleurs après réception des résultats de Bob et Charlie soit de 0.4, donc supérieure à la valeur obtenue par la représentation d'Eve (cf. figure 3). Il ne sera alors pas nécessaire d'interroger cette dernière : elle ne pourra pas avoir de résultat dans le k-meilleurs global. Mais cette condition ne pourra être atteinte que si la représentation est *optimiste*, c'est-à-dire que la valeur de pertinence de la représentation par rapport à la requête ne peut pas être dépassée par une des composantes de cette représentation.

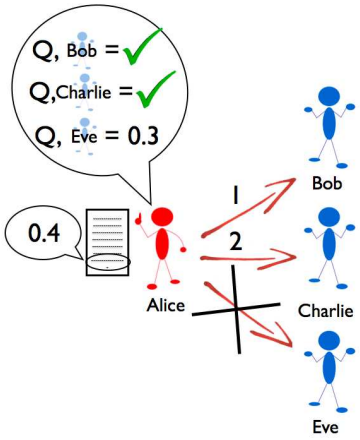

Fig. 3. Avec notre représentation optimiste, il n'est pas nécessaire d'interroger Eve.

L'exemple précédent est très simple car il ne considère que les voisins immédiats d'Alice. Il se généralise bien à condition que les pairs n'aient pas une représentation uniquement de leurs voisins mais de leur voisinage (voisins et voisins de leurs voisins jusqu'à un rayon donné). Il faut donc que la représentation puisse intégrer des ensembles de documents, mais aussi des ensembles de pairs, et si possible de la même façon. Ainsi il sera possible d'utiliser notre démarche pour tous les pairs interrogés avec un TTL de la requête égal à leur profondeur de connaissance du voisinage.

Reprenons l'exemple précédent en ajoutant des voisins à Charlie, Dave, et Eve, Franck (cf. figure 4). Si le TTL de la requête vaut 1, alors nous sommes dans le cas précédent et la représentation des voisins directs d'Alice permet à cette dernière de guider la requête, voir de couper des envois. Si le TTL est égal à 2, alors Alice considère la représentation de Charlie et Eve avec une profondeur de 1, tenant compte de Dave et Franck. Le processus précédent peut alors être utilisé. Bob conserve la plus forte valeur et est classé numéro un. C'est la représentation de {Charlie,Dave} qui vient ensuite, avec 0.6. Enfin la valeur la plus faible est celle de la représentation de {Eve,Franck}. Il est aussi possible de couper (cf. figure 5) ce dernier envoi si par exemple, après l'envoi à Bob et Charlie, et réception de leurs résultats, la valeur minimale du k-meilleurs est supérieure à la représentation de {Eve,Franck}.

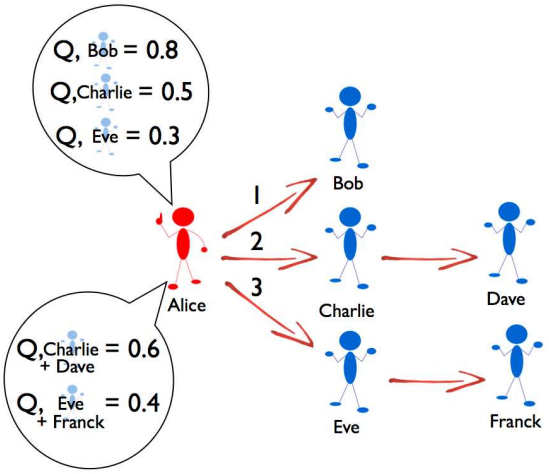

Fig. 4. Alice dispose d'une représentation de ses voisins à un rayon de 1 et à un rayon de 2. Suivant la valeur du TTL (1 ou 2), elle utilise l'un ou l'autre pour savoir dans quel ordre envoyer la requête.

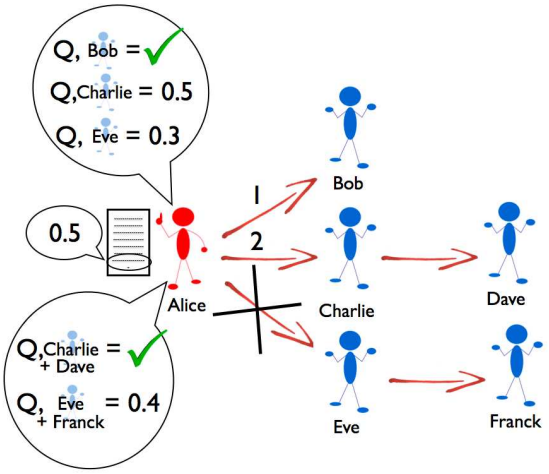

Fig. 5. Le TTL de la requête est à 2, Alice utilise la représentation avec un rayon de 2 et peut éviter l'envoi à Eve, si les représentations sont optimistes.

3 Agréger et concrétiser la représentation de documents pour aider le routage

Notre objectif est ici de définir les spécificités d'une représentation pouvant décrire un ensemble de documents, le contenu d'un pair, mais aussi un groupe

de pairs de manière optimiste. Dans la mesure où cette représentation doit être échangée entre pairs dans le système, elle se doit d'être synthétique : il n'est pas envisageable d'envoyer la représentation de tous les documents. Nous proposons donc d'introduire deux niveaux conceptuels différents (cf. figure 6). Outre le niveau des documents et des requêtes eux-même, nous introduisons le niveau des représentations ($\mathcal{R}$), celui où se situe les calculs de pertinence par exemple ; ainsi que le niveau des abstractions ($\mathcal{A}$ qui n'est pas forcément le même que $\mathcal{R}$), qui sont les descriptions d'ensembles de documents, de pairs, etc. Une fonction d'*agrégation* abstrait une agrégation à partir d'un ensemble de représentations de documents. Pour évaluer la pertinence à un même niveau d'abstraction, deux voies sont envisageables : (i) transformer la requête au même titre que les documents l'ont été ou (ii) à partir de l'agrégation, *concrétiser* une représentation particulière que l'on peut comparer à la requête. C'est cette deuxième solution que nous avons choisie, car elle permet de réutiliser directement la fonction d'évaluation de pertinence existante.

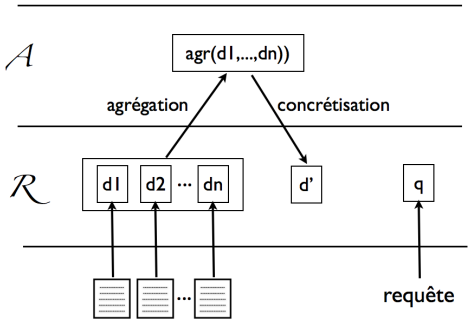

Fig. 6. Représentations de documents et requêtes (niveau 1). Les documents sont *abstraits* en une agrégation (niveau 2) qui peut être *concrétisée* pour être comparée à une requête au niveau des représentations.

Par abus de notation d et q n'indiqueront donc dans la suite pas un document ou une requête mais les *représentations* d'un document ou d'une requête.

Nous pouvons définir une fonction d'agrégation dans un cadre très général comme une fonction qui prend en entrée un ensemble d'éléments (par exemple des documents) et génère l'agrégation. Plus particulièrement, nous adoptons la définition suivante :

Définition 1 (Fonction d'agrégation)**.**
Soit $\mathcal{R}^{\mathcal{D}}$ l'ensemble des représentations de documents.
Une fonction est une fonction d'agrégation*, notée agr, si elle a la signature*

suivante :

$$agr : \left| \begin{array}{ll} \mathcal{R}^{\mathcal{D}} & \mapsto \mathcal{A} \\ \{d_1, \ldots, d_n\} & \rightarrow agr(\{d_1, \ldots, d_n\}) \end{array} \right.$$

Une fonction d'agrégation doit être une base commune à toutes les représentations. Elle doit décrire de manière synthétique les documents de la ou des collections qui la composent. C'est pourquoi nous voulons qu'elle soit facilement « extensible ». La première propriété envisagée est alors l'incrémentalité, c'est-à-dire le fait qu'elle puisse s'enrichir de nouveaux documents sans que l'on ait besoin de tout recalculer.

Définition 2 (Agrégation incrémentale)**.**
Soient $\{d_1, \ldots, d_n\}$, $n \in \mathbb{N}$ *un ensemble de documents et* d' *un document.*
Une agrégation agr est incrémentale *si :*
il existe une fonction

$$add : \left| \mathcal{A} \times \mathcal{R} \mapsto \mathcal{A} \right.$$

telle que
$\forall d' \notin \{d_1, \ldots, d_n\}$ $add(\{agr(\{d_1, \ldots, d_n\}), d'\}) = agr(\{d_1, \ldots, d_n, d'\})$

Il est facile de déduire de cette définition la commutativité (l'ordre d'ajout des documents n'importe pas) et l'incrémentalité ensembliste (plusieurs documents peuvent être ajoutés simultanément) d'une fonction d'agrégation incrémentale. Ces propriétés sont importantes pour simplifier la gestion de l'agrégation de plusieurs documents.

Nous pouvons définir ensuite la composabilité de la fonction d'agrégation, c'est-à-dire la possibilité de combiner deux agrégations.

Définition 3 (Agrégation composable)**.**
Soient $\{d_1^1, \ldots, d_{n_1}^1\}$ *et* $\{d_1^2, \ldots, d_{n_2}^2\}$, n_1 *et* $n_2 \in \mathbb{N}$ *deux ensembles de documents.*
Une agrégation agr est composable *si :*
il existe une fonction

$$merge : \left| \mathcal{A} \times \mathcal{A} \mapsto \mathcal{A} \right.$$

telle que
$merge(agr(\{d_1^1, \ldots, d_n^1\}), agr(\{d_1^2, \ldots, d_n^2\})) = agr(\{d_1^1, \ldots, d_n^1, d_1^2, \ldots, d_n^2\})$

Cette propriété permet de faciliter la gestion de l'agrégation de plusieurs ensembles de documents.

Une fonction de concrétisation est une fonction qui prend en entrée une abstraction d'éléments (par exemple de documents) et génère une représentation. Cette représentation ne correspond pas nécessairement à un document existant, mais simplement une description compatible avec l'abstraction dont elle est issue en rapport avec la requête.

Définition 4 (Fonction de concrétisation par rapport à une requête)**.**
Soit $\overrightarrow{q}$ *une requête.*

Une fonction, notée concr, est une fonction de concrétisation *si elle a la signature suivante :*
$concr : \begin{vmatrix} \mathcal{A} \mapsto \mathcal{R} \\ a \ \rightarrow concr_q(a) \end{vmatrix}$

Notons aussi que les mesures de pertinence ont pour arguments des éléments de $\mathcal{R}$, pas de $\mathcal{A}$. Nous souhaitons qu'une fonction de concrétisation soit *optimiste par rapport à une mesure de pertinence*, c'est-à-dire qu'elle ne sous-estime pas les valeurs de ses composantes par rapport à cette mesure.

Définition 5 (Concrétisation optimiste par rapport à une mesure de pertinence)**.**
Soient $\{d_1, \ldots, d_n\},\ n \in \mathbb{N}$ *un ensemble de documents et q une requête.*
Soit rel une mesure de pertinence entre éléments de $\mathcal{R}$*.*
concr est une concrétisation optimiste *si :*
$\forall i,\ 1 \leq i \leq n,\ rel(q, concr_q(agr(\{d_1, \ldots, d_n\}))) \geq rel(q, d_i)$.

4 Agrégation et concrétisation pour les vecteurs sémantiques

Dans cette section, nous supposons que documents et requêtes sont représentés par des vecteurs sémantiques. $\mathcal{R}$ est donc l'ensemble de tous les vecteurs sémantiques.

4.1 Représentation de documents et requêtes par vecteurs sémantiques

Le modèle vectoriel est un modèle classique en recherche d'information [4]. L'idée de vecteurs sémantiques est elle plus récente [21,11]. Formellement, cela consiste à prendre les concepts d'une ontologie comme dimensions d'un espace de représentation pour documents et requêtes. Le cadre général d'un tel modèle est celui de la figure 7 : requêtes et documents sont indexés sémantiquement et les documents sont classés par rapport à chaque requête, par exemple en utilisant le cosinus comme mesure de pertinence.

Définition 6 (Vecteur sémantique)**.**
Un vecteur sémantique $\overrightarrow{v_\Omega}$ *est une application définie sur l'ensemble des concepts* $\mathcal{C}_\Omega$ *d'une ontologie* $\Omega\ \forall c \in \mathcal{C}_\Omega, \overrightarrow{v_\Omega} : c \rightarrow [0,1]$

Considérer les valeurs dans $[0,1]$ est arbitraire et ne change en rien le raisonnement. Dans ce modèle vectoriel, les concepts de l'ontologie sont souvent appelés les dimensions des vecteurs. Par exemple, considérons l'ontologie de la figure 8. Elle est composée de douze concepts, avec des liens de subsomption (*is-a*) entre eux. Par exemple le concept *public school* est une sorte (*is-a*) du concept *school*. Sur le vecteur, nous voyons que $\overrightarrow{d_i}[bank] = 1$, $\overrightarrow{d_i}[financial\ institution] = 0.5$, $\overrightarrow{d_i}[central\ bank] = 0.8$, $\overrightarrow{d_i}[university] = 0.8$ et $\overrightarrow{d_i}[school] = 0.2$. Ce qui signifie

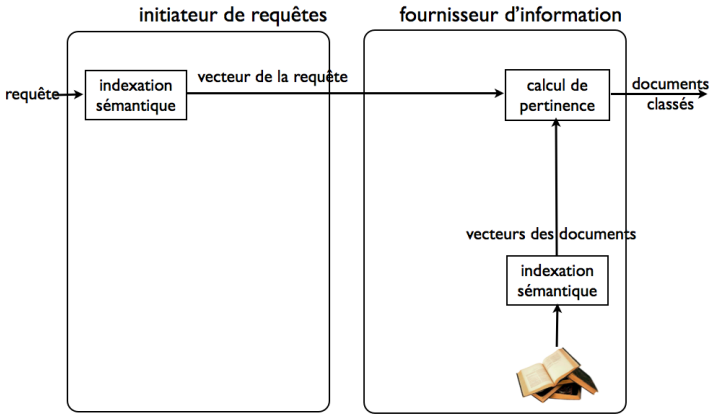

Fig. 7. Cadre d'un système de recherche d'information utilisant le modèle vectoriel sémantique.

que le document est lié à ces cinq dimensions, plus fortement au concept *bank* qu'aux autres, même si les concepts *central bank* et *university* sont importants. En ce qui concerne les deux autres concepts, *financial institution* et *school*, leur pondération indique que ce ne sont pas des dimensions centrales pour le document.

4.2 ALL@1, une fonction d'agrégation incrémentale et composable

Nous proposons la définition de ALL@1, une fonction d'agrégation satisfaisant les propriétés énoncées précédemment.

Définition 7 (ALL@1)**.**
Soient Ω une ontologie et $\mathcal{C}_\Omega$ l'ensemble de ses concepts.
Soit $\{\overrightarrow{d_1}, \ldots, \overrightarrow{d_n}\}$, $n \in \mathbb{N}$ un ensemble de documents sur $\mathcal{C}_\Omega$.
ALL@1 *est une fonction d'agrégation telle que :*

$$\forall c \in \mathcal{C}_\Omega \ all@1(\{\overrightarrow{d_1}, \ldots, \overrightarrow{d_n}\})[c] = \begin{cases} 1 \ si \ \exists \overrightarrow{d_i}, \ i \in [0..1], \ \overrightarrow{d_i}[c] > 0 \\ 0 \ sinon \end{cases}$$

ALL@1 est donc une fonction qui donne à la caractérisation d'un ensemble de documents la valeur 1 à chaque dimension pour laquelle il existe un document qui la pondère de manière non nulle. L'ensemble des agrégations $\mathcal{A}$ est donc l'ensemble des vecteurs sémantiques n'utilisant que des pondérations à 0 ou 1. Il en découle que $\mathcal{A} \subseteq \mathcal{R}$. Cette propriété permet d'utiliser ALL@1 en lieu et place

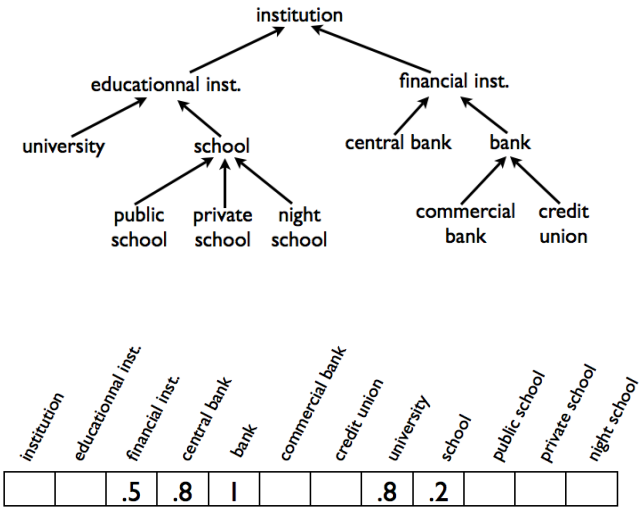

Fig. 8. Une ontologie restreinte, composée de douze concepts avec les liens de subsomption (*is-a*) et le vecteur sémantique d'un document caractérisé sur cette ontologie.

des fonctions *add* et *merge* pour exprimer les propriétés d'incrémentalité et de composabilité.

Théorème 1 (ALL@1 est incrémentale)**.**
Soient Ω une ontologie et $\mathcal{C}_\Omega$ l'ensemble de ses concepts.
Soit $\{\overrightarrow{d_1}, \ldots, \overrightarrow{d_n}\}$, $n \in \mathbb{N}$ un ensemble de documents sur $\mathcal{C}_\Omega$.
$\forall\, \overrightarrow{d} \notin \{\overrightarrow{d_1}, \ldots, \overrightarrow{d_n}\}\ \forall c \in \mathcal{C}_\Omega$ ALL@1$(\{all@1(\{\overrightarrow{d_1}, \ldots, \overrightarrow{d_n}\}), \overrightarrow{d}\}) = all@1(\{\overrightarrow{d_1}, \ldots, \overrightarrow{d_n}, \overrightarrow{d}\})$

Théorème 2 (ALL@1 est composable)**.**
Soient Ω une ontologie et $\mathcal{C}_\Omega$ l'ensemble de ses concepts.
Soient $\{\overrightarrow{d_1^1}, \ldots, \overrightarrow{d_{n_1}^1}\}$ et $\{\overrightarrow{d_1^2}, \ldots, \overrightarrow{d_{n_2}^2}\}$, n_1 et $n_2 \in \mathbb{N}$ deux ensembles de documents sur $\mathcal{C}_\Omega$.
ALL@1$(all@1(\{\overrightarrow{d_1^1}, \ldots, \overrightarrow{d_n^1}\}), all@1(\{\overrightarrow{d_1^2}, \ldots, \overrightarrow{d_n^2}\})) = all@1(\{\overrightarrow{d_1^1}, \ldots, \overrightarrow{d_n^1}, \overrightarrow{d_1^2}, \ldots, \overrightarrow{d_n^2}\})$

Les preuves de ces deux théorèmes sont triviales.

4.3 Adapt2Query, une fonction de concrétisation optimiste

Nous proposons la définition d'une fonction de concrétisation relative à une requête : Adapt2Query.

Définition 8 (ADAPT2QUERY)**.**
Soient Ω une ontologie et $\mathcal{C}_\Omega$ l'ensemble de ses concepts.
Soit $\{\overrightarrow{d_1}, \ldots, \overrightarrow{d_n}\}$, $n \in \mathbb{N}$ un ensemble de documents sur $\mathcal{C}_\Omega$. Soit $\overrightarrow{q}$ une requête sur $\mathcal{C}_\Omega$.
ADAPT2QUERY *, notée $A2Q_{\overrightarrow{q}}$, est une fonction de concrétisation telle que :*
$\forall c \in \mathcal{C}_\Omega \ A2Q_{\overrightarrow{q}}(all@1(\{\overrightarrow{d_1}, \ldots, \overrightarrow{d_n}\}))[c] = \overrightarrow{q}[c] \times all@1(all@1(\{\overrightarrow{d_1}, \ldots, \overrightarrow{d_n}\}))[c]$

Cette définition attribue à chaque dimension partagée entre la requête et l'ensemble de documents la pondération que lui confère la requête. ALL@1 est un masque booléen pour les vecteurs sémantiques des requêtes, qui permet de manière efficace de conserver les dimensions apparaissant dans une collection et de générer ADAPT2QUERY.

Théorème 3 (ADAPT2QUERY est une fonction de concrétisation optimiste par rapport au cosinus)**.**
Soient Ω une ontologie et $\mathcal{C}_\Omega$ l'ensemble de ses concepts.
Soit $\{\overrightarrow{d_1}, \ldots, \overrightarrow{d_n}\}$, $n \in \mathbb{N}$ un ensemble de documents quelconque sur $\mathcal{C}_\Omega$. Nous avons : $\forall i, \ 1 \leq i \leq n \ \cos(\overrightarrow{q}, A2Q_{\overrightarrow{q}}(all@1(\{\overrightarrow{d_1}, \ldots, \overrightarrow{d_n}\}))) \geq \cos(\overrightarrow{q}, \overrightarrow{d_i})$

Démonstration.
Notons $\overrightarrow{a} = \text{A2Q}_{\overrightarrow{q}}(\text{all@1}(\{\overrightarrow{d_1}, \ldots, \overrightarrow{d_n}\}))$.
Supposons qu'il existe un document $\overrightarrow{d_i}$, $i \in [1..n]$ dans l'agrégation qui ait un cosinus avec la requête supérieur à celui de la représentation avec la requête $\overrightarrow{q}$. Supposons que la représentation ait k concepts en commun avec la requête, et que nous ayions $|\mathcal{C}_{\overrightarrow{q}}| = l$, avec $\mathcal{C}_{\overrightarrow{q}}$ représentant l'ensemble des concepts centraux de la requête $\overrightarrow{q}$. Dans ce cas, $l \geq k$. Supposons que $\overrightarrow{d_i}$ ait j concepts en commun avec la requête. Par définition, nous savons que $\{c_1, \ldots c_j\} \subseteq \{c_1, \ldots c_k\} \subseteq \{c_1, \ldots c_l\}$, car les concepts du documents pondèrent ALL@1, qui permet d'avoir une pondération dans ADAPT2QUERY, seulement si la requête en a une aussi : le document ne peut donc pas avoir plus de concepts en commun avec la requête de que l'agrégation.

$$\cos(\overrightarrow{q}, \overrightarrow{a}) < \cos(\overrightarrow{q}, \overrightarrow{d_i})$$
$$\frac{\overrightarrow{q} \cdot \overrightarrow{a}}{|\overrightarrow{q}| \times |\overrightarrow{a}|} < \frac{\overrightarrow{q} \cdot \overrightarrow{d_i}}{|\overrightarrow{q}| \times |\overrightarrow{d_i}|}$$
$$\frac{\overrightarrow{q} \cdot \overrightarrow{a}}{|\overrightarrow{a}|} < \frac{\overrightarrow{q} \cdot \overrightarrow{d_i}}{|\overrightarrow{d_i}|}$$

Or, puisque $\{c_1, \ldots c_k\} \subseteq \{c_1, \ldots c_l\}$n nous avons donc :

$$\frac{\sum_{c \in \{c_1, \ldots c_k\}} \overrightarrow{q}[c] \times \overrightarrow{a}[c]}{\sqrt{\sum_{c \in \{c_1, \ldots c_k\}} \overrightarrow{a}[c]^2}} < \frac{\overrightarrow{q} \cdot \overrightarrow{d_i}}{|\overrightarrow{d_i}|}$$

Et puisque par construction $\overrightarrow{a}[c] = \overrightarrow{q}[c]$ pour tous les concepts qu'ils ont en commun :

$$\frac{\sum_{c\in\{c_1,\dots c_k\}} \overrightarrow{a}[c]^2}{\sqrt{\sum_{c\in\{c_1,\dots c_k\}} \overrightarrow{a}[c]^2}} < \frac{\overrightarrow{q}\cdot\overrightarrow{d_i}}{|\overrightarrow{d_i}|}$$

Or, $\forall x \in \mathbb{R}\ \frac{x}{\sqrt{x}} = \sqrt{x}$. Donc :

$$\sqrt{\sum_{c\in\{c_1,\dots c_k\}} \overrightarrow{a}[c]^2} < \frac{\sum_{c\in\{c_1,\dots c_j\}} \overrightarrow{q}[c]\times\overrightarrow{d_i}[c]}{\sqrt{\sum_{c\in\mathcal{C}_\Omega} \overrightarrow{d_i}[c]^2}}$$

Encore une fois, par définition, les valeurs de $\overrightarrow{a}$ et de $\overrightarrow{q}$ sont identiques pour tous les concepts qu'ils ont en commun, $\{c_1,\dots c_k\}$, et comme $\{c_1,\dots c_j\} \subseteq \{c_1,\dots c_k\}$:

$$\sqrt{\sum_{c\in\{c_1,\dots c_k\}} \overrightarrow{a}[c]^2} < \frac{\sum_{c\in\{c_1,\dots c_j\}} \overrightarrow{a}[c]\times\overrightarrow{d_i}[c]}{\sqrt{\sum_{c\in\mathcal{C}_\Omega\}} \overrightarrow{d_i}[c]^2}}$$

$$1 < \frac{\sum_{c\in\{c_1,\dots c_j\}} \overrightarrow{a}[c]\times\overrightarrow{d_i}[c]}{\sqrt{\sum_{c\in\{c_1,\dots c_k\}} \overrightarrow{a}[c]^2}\times\sqrt{\sum_{c\in\mathcal{C}_\Omega} \overrightarrow{d_i}[c]^2}}$$

$$1 < \cos(\overrightarrow{a},\overrightarrow{d_i})$$

Ce qui n'est pas possible, le cosinus prenant ses valeurs dans $[0,1]$. Il n'y a donc pas de document pouvant avoir une valeur de pertinence supérieure à celle de la représentation ADAPT2QUERY. ADAPT2QUERY est une fonction de représentation optimiste. □

5 Cadre sémantiquement hétérogène : quelques réflexions

Nous considérons qu'un système est sémantiquement hétérogène lorsque différentes ontologies peuvent être utilisées par différents pairs. Le concept de k meilleurs documents reste le même à condition d'accepter que chaque pair est responsable de l'évaluation de la pertinence de ses document et de faire l'hypothèse que chacun évalue la pertinence dans le même référentiel, i.e. le même intervalle. Même sous ces hypothèses, le problème majeur de l'hétérogénéité reste la compréhension d'une requête.

Une première solution consiste à mettre en place des alignements entre ontologies, c'est-à-dire des correspondances entre les concepts et relations de ces ontologies [7]. Bien souvent, parmi celles-ci, seules sont considérées les équivalences entre concepts. Dans ce cas, la compréhension d'une requête est réduite aux concepts partagés qu'elle pondère. Certains concepts de la requête sont donc "perdus" pour le fournisseur de documents. Cela peut altérer fortement sa compréhension de la demande. Néanmoins, l'agrégation des vecteurs sémantiques

décrivant ses documents peut être définie comme dans le cas homogène présenté précédemment, avec les mêmes propriétés.

Pour assurer une meilleure compréhension entre initiateur et fournisseurs, nous avons proposé une méthode qui permet d'utiliser des concepts non partagés pour exprimer la requête (côté initiateur) et pour l'interpréter (côté fournisseur) [20].

L'architecture générale entre deux pairs est celle présentée à la figure 9. Après indexation de la requête dans son ontologie, l'initiateur en génère une expansion qui explique les différentes dimensions de celle-ci. S'appuyant sur les équivalences entre leurs deux ontologies, le fournisseur interprète les explications dans sa propre ontologie. Il s'appuie sur son interprétation pour adapter la description de chacun de ses documents à la requête. Soulignons que l'architecture ne nécessite aucune modification des modules d'un système d'information distribué homogène sémantiquement (cf. figure 7, section 2), et que les nouveaux modules s'intègrent facilement. Cette solution a été appelée ExSI^{2}D pour Expansion Structurante, Interprétation et Image des Documents.

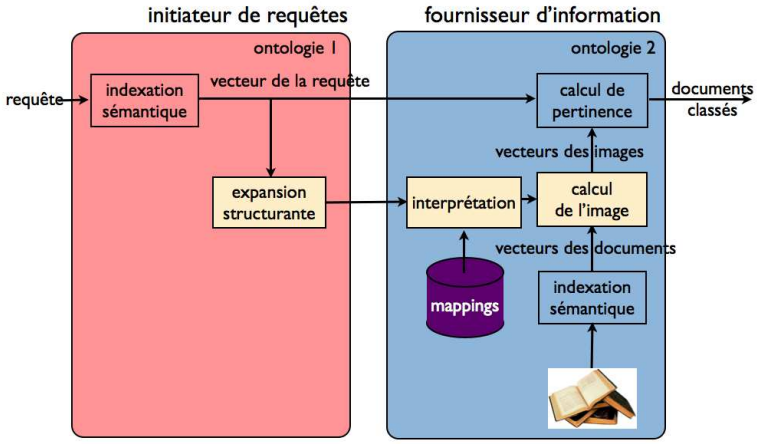

Fig. 9. Le cadre d'ExSI^{2}D.

L'agrégation de vecteurs sémantiques par un pair est toujours possible de la même manière que précédemment. La phase de concrétisation doit être adaptée de sorte à prendre en compte le fait qu'ExSI^{2}D évalue la pertinence de la requête non pas par rapport au vecteur sémantique d'un document, mais par rapport à son image. Le détail de cette fonction de concrétisation peut être trouvé dans [20].

Elle reste optimiste tant que l'on considère un seul pair ou des pairs utilisant la même ontologie. Elle permet à un pair de déterminer rapidement s'il dispose potentiellement de documents dépassant un certain seuil de pertinence sans avoir à analyser toute sa collection.

L'agrégation de vecteurs sémantiques représentés dans des espaces différents (ontologies différentes) est un problème en soi. La solution consistant à expliquer et interpréter une agrégation est envisageable, mais nous n'avons pas été en mesure de proposer une concrétisation optimisite dans ce cas.

6 Travaux liés

Certains algorithmes de routage dans les systèmes P2P non structurés essaient d'avoir un aperçu de leur voisinage pour diriger les envois vers les voisins statistiquement plus intéressants [10,18,5,22]. Il faut tout d'abord noter que contrairement à ces approches, nous ne nous limitons pas à des requêtes *exact match*, c'est-à-dire avec une fonction de pertinence booléenne, mais que nous pouvons aussi faire du routage sur des requêtes k-meilleurs. Les autres solutions peuvent sans doute s'adapter au cadre k-meilleurs, mais sans doute difficilement, car il faudrait avoir une fonction d'évaluation d'une liste de résultats retournée à une requête. D'autre part, même si nous partageons le même objectif que ces techniques, qui est de guider les requêtes vers les meilleurs pairs, nous avons aussi la possibilité de ne pas envoyer aux pairs ayant des résultats médiocres tout en étant sûrs que c'est vrai (optimisme).

De nombreuses solutions essaient de générer des correspondances entre pairs dans un système distribué hétérogène sémantiquement. Dans SomeWhere [15] ou Piazza [9], les pairs créent des mappings entre leurs ontologies et réécrivent les requêtes lors du routage. D'autres solutions essaient de trouver de manière semi-automatique des mappings, comme Hyperion [14] ou GLUE [6], grâce à des tables de mappings et l'aide de l'utilisateur. Quelques systèmes essaient même de découvrir automatiquement les mappings (PeerDB [13], Chatty Web [1]) qui servent ensuite à diriger le routage vers les pairs qui peuvent comprendre les requêtes. Notre solution ne s'oppose pas à ces approches qui cherchent à construire des correspondances entre ontologies, elle les prolonge. Dans toutes ces approches, les alignements ne sont pas complets et il demeure des parties non partagées entre les ontologies. Il est alors possible d'utiliser ExSI^{2}D pour gagner en interopérabilité.

Dans le domaine des bases de données, de nombreux travaux concernent l'obtention de modèles de haut-niveau. Les résumés [16] permettent d'obtenir une réponse, de naviguer, etc. sans forcément interroger l'ensemble de la base de données. Ce sont des représentations toutefois assez volumineuses et dont la construction et la mise à jour peuvent être coûteuses. De plus, elles ne sont en général pas pensées dans un objectif de routage de requêtes.

Les solutions de clustering et classification des documents [17] utilisent ou peuvent générer des abstractions pour le modèle vectoriel (centroïdes, médoïdes, ...). Néanmoins, ces abstractions ne sont pas utilisées avec le même objectif que

nous le faisons avec nos représentations. Pour la classification, les abstractions peuvent servir à estimer la classe à laquelle doit appartenir un document, ou si une classe doit évoluer (fusionner, se scinder, etc.). S'il peut aussi servir pour le routage de requêtes, il n'est cependant pas optimiste, ce qui le rend évidemment moins utile que notre solution.

7 Conclusion

Dans cet article nous avons proposé de distinguer fonctions d'agrégation et de concrétisation. La première permet d'agréger un ensemble de représentations d'informations. On passe ainsi à un niveau d'abstraction plus haut. La deuxième permet de repasser dans le même espace de représentation que celui des informatinos, ce qui permet de confronter le résultat d'une agrégation à la représentation d'une requête. Nous avons défini les propriétés d'incrémentalité et composabilité d'une agrégation qui permettent une gestion simplifiée des regroupements dans un système distribué. La propriété d'optimisme d'une fonction de concrétisation a été proposée. Elle présente l'avantage d'assurer qu'aucune information ne peut avoir une mesure de pertinence supérieure à celle obtenue par une agrégation dans laquelle elle apparait. Nous avons montré sur un certain nombre d'exemples l'intérêt que cela peut avoir pour guider la recherche d'une information dans un système distribué, en permettant en particulier d'éviter de visiter certains pairs. Cette propriété peut aussi être utilisée par un pair localement pour éviter de consulter systématiquement l'ensemble de sa collection d'informations.

Dans le cadre d'une représentation par vecteurs sémantiques, nous avons proposé une agrégation incrémentale et composable ainsi qu'une concrétisation optimiste par rapport à la mesure de pertinence usuelle qu'est le cosinus. Ces définitions et propriétés peuvent être utilisées dans un cadre distribué sémantiquement homogène. Nous avons montré qu'un cadre hétérogène présente un certain nombre de difficultés liées à l'utilisation d'espaces de représentation différents.

Ces travaux peuvent être étendus dans deux directions. La première concerne l'étude de propriétés supplémentaires de l'abstraction et de la concrétisation. En particulier, nous n'avons pour l'instant pas considéré le retrait d'une information. Prendre en compte au plus vite, et donc à un coût minimal, un retrait d'information nous a semblé moins important car si un retrait n'est pas reflété dans l'agrégation on conserve toutefois la propriété d'optimisme.

La deuxième direction concerne l'exploitation de l'agrégation et de la concrétisation par les algorithmes de recherche top-k. En premier lieu, une évaluation précise des informations accessibles via les voisins permet de guider l'envoi de la requête. En second lieu, la propriété d'optimisme permet d'éviter l'envoi de la requête aux voisins via lesquels il est alors certain que ne sera obtenue aucune information plus pertinente que celles déjà obtenues. Il semble donc nécessaire de revisiter un certain nombre d'algorithmes qui guideraient ainsi mieux la recherche.

Références

1. K. Aberer, P. Cudré-Mauroux, and M. Hauswirth. The chatty web : emergent semantics through gossiping. In *WWW '03 : Proceedings of the 12th international conference on World Wide Web*, pages 197–206, New York, NY, USA, 2003. ACM.
2. R. Akbarinia, E. Pacitti, and P. Valduriez. Reducing network traffic in unstructured p2p systems using top-k queries. *Distributed and Parallel Databases*, 19(2-3) :67–86, 2006.
3. R. Akbarinia, E. Pacitti, and P. Valduriez. Query processing in p2p systems. Technical Report 6112, INRIA, France, 2007.
4. M. W. Berry, Z. Drmac, and E. R. Jessup. Matrices, vector spaces, and information retrieval. *SIAM Rev.*, 41(2) :335–362, 1999.
5. A. Crespo and H. Garcia-Molina. Routing indices for peer-to-peer systems. In *ICDCS '02 : Proceedings of the 22 nd International Conference on Distributed Computing Systems (ICDCS'02)*, page 23, Washington, DC, USA, 2002. IEEE Computer Society.
6. A. Datta, M. Hauswirth, and K. Aberer. Updates in highly unreliable, replicated peer-to-peer systems. In *ICDCS '03 : Proceedings of the 23rd International Conference on Distributed Computing Systems*, page 76, Washington, DC, USA, 2003. IEEE Computer Society.
7. J. Euzenat and P. Shvaiko. *Ontology matching*. Springer-Verlag, Heidelberg (DE), 2007.
8. L. Gravano and A. Marian. Optimizing top-k selection queries over multimedia repositories. *IEEE Trans. on Knowl. and Data Eng.*, 16(8) :992–1009, 2004. Member-Surajit Chaudhuri.
9. A. Y. Halevy, Z. G. Ives, P. Mork, and I. Tatarinov. Piazza : data management infrastructure for semantic web applications. In *WWW '03 : Proceedings of the 12th international conference on World Wide Web*, pages 556–567, New York, NY, USA, 2003. ACM.
10. V. Kalogeraki, D. Gunopulos, and D. Zeinalipour-Yazti. A local search mechanism for peer-to-peer networks. In *CIKM '02 : Proceedings of the eleventh international conference on Information and knowledge management*, pages 300–307, New York, NY, USA, 2002. ACM.
11. R. Krovetz and W. B. Croft. Lexical ambiguity and information retrieval. *Information Systems*, 10(2) :115–141, 1992.
12. S. Michel, P. Triantafillou, and G. Weikum. Klee : a framework for distributed top-k query algorithms. In *VLDB '05 : Proceedings of the 31st international conference on Very large data bases*, pages 637–648. VLDB Endowment, 2005.
13. B. C. Ooi, Y. Shu, and K.-L. Tan. Relational data sharing in peer-based data management systems. *SIGMOD Rec.*, 32(3) :59–64, 2003.
14. P. Rodríguez-Gianolli, A. Kementsietsidis, M. Garzetti, I. Kiringa, L. Jiang, M. Masud, R. J. Miller, and J. Mylopoulos. Data sharing in the hyperion peer database system. In *VLDB '05 : Proceedings of the 31st international conference on Very large data bases*, pages 1291–1294. VLDB Endowment, 2005.
15. M.-C. Rousset. Small can be beautiful in the semantic web. In *International Semantic Web Conference*, pages 6–16, 2004.

16. R. Saint-Paul, G. Raschia, and N. Mouaddib. General purpose database summarization. In *VLDB*, pages 733–744, 2005.

17. M. Steinbach, G. Karypis, and V. Kumar. A comparison of document clustering techniques. In *Proc. of Text Mining Workshop, SIGKDD*, 2000.

18. D. Tsoumakos and N. Roussopoulos. Adaptive probabilistic search for peer-to-peer networks. In *P2P '03 : Proceedings of the 3rd International Conference on Peer-to-Peer Computing*, pages 102–109, Washington, DC, USA, 2003. IEEE Computer Society.

19. D. Tsoumakos and N. Roussopoulos. Analysis and comparison of p2p search methods. In *InfoScale '06 : Proceedings of the 1st international conference on Scalable information systems*, page 25, New York, NY, USA, 2006. ACM.

20. A. Ventresque, S. Cazalens, P. Lamarre, and P. Valduriez. Improving interoperability using query interpretation in semantic vector spaces. In *ESWC'08, European Semantic Web Conference*, pages 539–553, 2008. nominee for best paper award (4/51 accepted papers).

21. W. Woods. Conceptual indexing : A better way to organize knowledge. Technical report, Sun Microsystems Laboratories, April 1997.

22. B. Yang and H. Garcia-Molina. Improving search in peer-to-peer networks. In *ICDCS '02 : Proceedings of the 22 nd International Conference on Distributed Computing Systems (ICDCS'02)*, page 5, Washington, DC, USA, 2002. IEEE Computer Society.